"Express Mail" mailing label number:

EV 335895926 US

SYSTEM AND METHOD FOR FACILITATING XML ENABLED IMS TRANSACTIONS

Field of the Disclosure

[0001] The present disclosure relates generally to computer software, and more specifically to IMS software.

Background

[0002] A significant portion of corporate data in the United States and abroad resides on mainframe computers, e.g., S/390 mainframes manufactured by International Business Machines. Much of the information stored on mainframe computers is managed using information management systems (IMS).

[0003] Typically, an IMS includes an IMS connect program and an IMS application program that can communicate with each other when a request is submitted to the IMS connect program. Many current IMS customers require access to IMS applications from z/OS and non-z/OS environments using XML. As such, extensible markup language (XML) is becoming a core technology to IMS applications. Thus, there exists a need for XML support in IMS Connect programs.

[0004] Accordingly, there is a need for a system and method for facilitating XML enabled IMS transactions.

Summary

[0005] A method for facilitating XML enabled IMS transactions includes receiving an XML input request at an IMS connect program and creating an input request byte array from the XML input request within the IMS connect program. Thereafter, the input request byte array is transmitted from the IMS connect program to an IMS application program.

[0006] In a particular embodiment, an output response byte array is generated within the IMS application program. The output response byte array is transmitted to the IMS connect program. An XML output response is created from the output response byte array within the IMS connect program, and the XML output response is transmitted to a user computer connected to the IMS connect program.

[0007] Moreover, in a particular embodiment, the IMS connect program includes a XML processor, and the method further includes transmitting the XML input request to a queue header within the XML processor. An XML server within the XML processor retrieves an XML input request control block from the queue header. Moreover, an XML adapter routine is invoked within the IMS connect program. The XML input request can be parsed and translated to create an input request byte array. Thereafter, the input request byte array is transmitted to the XML server. The XML server transmits the input request byte array to an IMS application program. Based on the input request byte array, an output response byte array is generated within the IMS application program.

[0008] In a particular embodiment, the output response byte array is transmitted to the queue header within the XML processor. The XML server within the XML processor retrieves an output response control block from the queue header. Further, an XML adapter routine is invoked within the IMS connect program and the output response byte array is parsed and translated to create an XML output response. The XML output response is transmitted to the user computer.

[0009] In another aspect of the illustrative embodiment of the present invention, a system for facilitating XML enabled IMS transactions includes a mainframe server. An IMS connect program and an IMS application program reside in the mainframe server. The IMS application program communicates with the IMS connect program. In this embodiment of the present invention, the IMS connect program includes logic for receiving at least one XML input request. The IMS connect program creates an input request byte array from the XML input request and transmits the input request byte array to an IMS application program.

[0010] In yet another aspect a computer program device is disclosed for facilitating XML enabled IMS transactions between a user computer and an IMS application program. The computer program device includes logic for receiving an XML input request from the user computer. The computer program device creates an input request byte array from the XML input request and transmits the input request byte array to an IMS application program.

Brief Description of the Drawings

[0011] FIGURE 1 is a block diagram of a system for facilitating XML enabled IMS transactions;

[0012] FIGURE 2 is a flow chart to illustrate operating logic of the system of FIGURE 1; and

[0013] FIGURE 3 is a flow chart to illustrate a method for facilitating XML enabled IMS transactions.

Description of the Drawings

[0014] Referring initially to FIGURE 1, an information management system (IMS) is shown and is generally designated 10. As shown, the system 10 includes a mainframe server 12 in which an operating system, e.g., OS/390 or zOS, is installed. At least one user computer 14 can access the mainframe server 12 via a web server 16. As shown, plural servlets 18, e.g., Java programs, reside on the web server 16. Additionally, one or more roll-your-own (RYO) clients 20 can access the mainframe server 12 via transmission control protocol/internet protocol (TCP/IP). It is to be understood that RYO clients 20 are those clients that provide their own software to interact with the mainframe server 12.

[0015] FIGURE 1 further shows that the mainframe server 12 includes an IMS connect program 24 that communicates with an IMS application program 26. It is to be understood that the IMS connect program 24 provides communication linkages between TCP/IP clients, e.g., via one or more user computers 14, and the IMS application program 26. The IMS connect program 24 can also provide communication linkages between one or more RYO clients 20 and the IMS application program 26. During operation, as described in detail below, the IMS connect program 24 communicates with the IMS application program 26 via a byte array.

[0016] As shown in FIGURE 1, the IMS connect program 24 includes an extensible markup language (XML) processor 28 that includes plural XML servers 30 and plural queue headers 32. The XML processor 28 communicates with a port task control block (TCB) 34 and with several plugins. In an alternative embodiment, the plugins can include an XML initialization routine 36, an XML adapter routine 38, and an XML terminator routine 40. As shown, the XML adapter routine 38 can communicate with one or more data transformers. In the non-limiting exemplary embodiment shown in FIGURE 1, the XML adapter routine 38 can communicate with a PL/I transformer 42, a Cobol transformer 44, a C transformer 46, a message format services (MFS) transformer 48, a High Level Assembler (HLASM) transformer 50, and a RYO transformer 52.

FIGURE 1 also shows that the transformers 42 - 52 can communicate with an XML metadata interchange (XMI) repository 54.

[0017] Referring still to FIGURE 1, the IMS application program 26 includes a control region 56 that receives a byte array from the IMS connect program 24. The IMS application program 26 also includes a transactional application program 58. It is to be understood that the transactional application program 58 includes a message processing program (MPP) region, an interactive fast path (IFP) region, and a batch message processing (BMP) region. Further, it is to be understood that the IMS application program 26 is the area in the mainframe server 12 in which user queries or requests are processed in order to determine the corresponding outputs that can be returned via the IMS connect program 24. For example, a user can submit a query or request regarding a bank account, e.g., an online bank statement, in XML to the IMS connect program 24. The IMS connect program 24 can process that request, as described in detail below, and then, submit a processed request to the IMS application program 26, e.g., in a byte array. Further, the IMS application program 26 receives the request, determines the appropriate response to the request, and generates an output response that is returned to the IMS connect program 24, e.g., in a byte array. The IMS connect program 24 can then process the output response, as described in detail below, to create an XML response that, in turn, can be returned to the user computer 14 from which the initial response was received.

[0018] It is to be understood that in the system 10 described above, the logic of the present disclosure can be contained on a data storage device with a computer readable medium, such as a computer diskette. Or, the instructions may be stored on a magnetic tape, hard disk drive, electronic read-only memory (ROM), optical storage device, or other appropriate data storage device or transmitting device thereby making a computer program product, i.e., an article of manufacture. In an illustrative embodiment, the computer-executable instructions may be lines of C++ compatible code.

[0019] The flow charts herein illustrate the structure of the logic as embodied in computer program software. Those skilled in the art will appreciate that the flow charts illustrate the structures of computer program code elements including logic circuits on an integrated circuit. An implementation includes a machine component that renders the program elements in a form that instructs a digital processing apparatus (e.g., a computer) to perform a sequence of function steps corresponding to those shown.

[0020] Referring now to FIGURE 2, a particular embodiment of the overall operating logic is shown and commences at block 100 with a do loop wherein during IMS connect startup time, the succeeding steps are performed. At block 102, an XML processor environment is established. Next, at block 104, initialization is performed. Moving to block 106, the XMI repository 54 (FIGURE 1) is opened. Thereafter, at decision diamond 108, it is determined whether interpretive marshalling is required. If so, the logic proceeds to block 110 and metadata is loaded. The logic then continues to block 112 and when the IMS connect environment is closed, the XML terminator routine 40 (FIGURE 1) is called. Returning to decision diamond 108, if interpretive marshalling is not required, the logic proceeds directly to block 112. Then, at block 114, the XMI repository 54 (FIGURE 1) is closed. Moving to block 116, memory used for XML processing is released. Thereafter, logic ends at state 118.

[0021] Referring now to FIGURE 3, a method for facilitating XML enabled IMS transactions is shown. At block 150, an IMS transaction input request is received at the mainframe server 12 (FIGURE 1), e.g., at the IMS connect program 24 (FIGURE 1). Next, at decision diamond 152, it is determined whether the IMS transaction input request includes XML data. If not, the logic ends at state 154. If the IMS transaction input request does include XML data, the logic moves to decision diamond 156 and it is determined whether the input request needs to be processed using the IMS connect program 24 (FIGURE 1). If not, the logic ends at state 154. On the other hand, if the input request must be processed using the IMS connect program 24 (FIGURE 1), the

logic moves to block 158 and the input request is sent to one of the queue headers 32 (FIGURE 1).

[0022] Continuing to block 160, one of the XML servers 30 (FIGURE 1) within the IMS connect program 24 (FIGURE 1), retrieves the input request control block from the queue header 32 (FIGURE 1). Thereafter, at block 162, the XML server 30 (FIGURE 1) invokes the XML adapter routine 38 (FIGURE 1). Proceeding to block 164, the XML input request is sent to the XML adapter routine 38 (FIGURE 1). At block 166 it is determined which transformer, e.g., the PL/I transformer 42 (FIGURE 1), the COBOL transformer 44 (FIGURE 1), the C transformer 46 (FIGURE 1), the MFS transformer 48 (FIGURE 1), the HLASM transformer 50 (FIGURE 1), or the RYO transformer 52 (FIGURE 1), needs to be invoked in order to parse and translate the XML input request. Moving to block 168, the XML input request is parsed and transformed to create a byte array. Next, at decision diamond 170, it is determined whether the transformation process undertaken at block 168 is successful. If not, the logic moves to block 172 and an XML fault message is generated. Thereafter, the logic proceeds to block 174 and the XML fault message is returned as a response to the IMS connect client. The logic then ends at state 154.

[0023] Returning to decision diamond 170, if the transformation process at block 168 is successful, the logic moves to block 176. At block 176, when the appropriate transformer completes construction of the byte array, the byte array is returned to the XML server 30 (FIGURE 1). Proceeding to block 178, the XML server 30 (FIGURE 1) sends the byte array to the IMS application program 26 (FIGURE 1), e.g., to the transactional application program 58 (FIGURE 1) therein.

[0024] Still referring to FIGURE 3, at block 180, an output response from the IMS application program 26 is received by a client thread running under the Port TCB 34 (FIGURE 1). The client thread running under the Port TCB 34 (FIGURE 1) retrieves the control block from the output response at block 182. Moving to decision diamond 184, it

is determined whether the output response must be processed using the IMS connect program 24 (FIGURE 1). If not, the logic proceeds to block 174 and the response is returned as is to the IMS connect client, e.g., to a user computer 14 (FIGURE 1). At decision diamond 184, if the output response must be processed using the IMS connect program 24 (FIGURE 1), the logic continues to block 186 and the client thread running under the Port TCB 34 (FIGURE 1) sends the output response to one of the queue headers 32 (FIGURE 1). Thereafter, an XML server 30 (FIGURE 1) retrieves the output response control block from the queue header 32 (FIGURE 1) at block 188.

[0025] Continuing the description of the logic, at block 190, the XML adapter routine 38 (FIGURE 1) is invoked to process the output response. At block 192, it is determined which transformer, e.g., the PL/I transformer 42 (FIGURE 1), the COBOL transformer 44 (FIGURE 1), the C transformer 46 (FIGURE 1), the MFS transformer 48 (FIGURE 1), the HLASM transformer 50 (FIGURE 1), or the RYO transformer 52 (FIGURE 1), needs to be invoked to parse and translate the output response. Then, at block 194, the output response byte array is transformed to an XML output response. Moving to decision diamond 196, it is determined whether the transformation process undertaken at block 194 is successful. If not, the logic moves to block 172 and an XML fault message is generated. Thereafter, the logic proceeds to block 174 and the XML fault message is returned as a response to the IMS connect client. The logic then ends at state 154. Returning to decision diamond 196, if the transformation process at block 168 is successful, the logic moves to block 174 where an XML output response is returned to the IMS connect client. The logic then ends at state 154.

[0026] With the configuration of structure described above, it is to be appreciated that the system and method described above provides a means for facilitating XML enabled IMS transactions. The present disclosure provides a generic XML processor inside an IMS connect program to facilitate any TCP/IP clients, including WebSphere and non-WebSphere, to send and receive XML documents to and from existing IMS transaction business logic. Translations between XML documents and IMS transaction message data

structures occur within the IMS connect program under an XML task to parse and transform XML requests and responses. Further, the generic XML processor within the IMS connect program can provide data translation for both non-formatted and formatted IMS transactional messages in XML. Non-formatted XML messages can be, for example, COBOL, PL/I, C, or Java payload transactional data. Formatted XML messages are MFS-based XML messages. The present system can also allow a third-party provided data transformer to be plugged into the system to facilitate XML parsing and transformation.

[0027] While a particular embodiment of a SYSTEM AND METHOD FOR FACILITATING XML ENABLED IMS TRANSACTIONS has been illustrated and described in detail, it is to be understood that the disclosed embodiment of the present invention is representative of the subject matter which is broadly contemplated by the present invention, that the scope of the present invention fully encompasses other embodiments and that the scope of the present invention is accordingly to be limited by nothing other than the appended claims, in which reference to an element in the singular is not intended to mean "one and only one" unless explicitly so stated, but rather "one or more." All structural and functional equivalents to the elements of the above-described embodiment that are known or later come to be known to those of ordinary skill in the art are expressly incorporated herein by reference and are intended to be encompassed by the present claims. Moreover, it is not necessary for a device or method to address each and every problem sought to be solved by the present invention, for it is to be encompassed by the present claims. Furthermore, no element, component, or method step in the present disclosure is intended to be dedicated to the public regardless of whether the element, component, or method step is explicitly recited in the claims. No claim element herein is to be construed under the provisions of 35 U.S.C. section 112, sixth paragraph, unless the element is expressly recited using the phrase "means for."